UNIVERSITATEA POLITEHNICA TIMISOARA

Facultatea de Automatică și Calculatoare

Specializarea Calculatoare și Tehnologia Informatiei

SISTEME ÎNCORPORATE

IMPLEMENTARE SISTEM DE MONITORIZARE A NUMĂRULUI DE LOCURI LIBERE ÎNTR-O PARCARE

Studenți: Bîtea Vlad-Nicolae, Cocoreanu David Nicholas

Anul universitar: 2023-2024

*Tema proiectului:*

Caracteristici:

• Se vor utiliza fotodiode sau senzori de detecţie a obiectelor reflectorizante („reflective object sensors”) sau senzori cu ultrasunete pentru detectarea maşinilor care intră si ies. Se pot folosi alte tipuri de senzori conform cu decizia proiectantului.

• Modalitatea de legare a senzorilor va cuprinde cât mai puţine fire (se vor alege interfeţe seriale, precum I2C, CAN, LIN etc).

• Se va asigura afişarea numărului de locuri libere din parcare (afişaje cu segmente sau matrice de LED-uri sau afişaj LCD).

• Numărul iniţial al locurilor libere de parcare va fi prestabilit.

• Aplicaţia va dispune de 2 LED-uri: unul din ele va fi pornit atâta vreme cât mai există locuri libere, al doilea va fi pornit când nu vor mai fi locuri disponibile în cadrul parcării. Nu este posibilă pornirea concomitentă a celor două LED-uri.

• Demonstrarea practică a proiectului va include o machetă cu un anumit număr de locuri de parcare delimitate, iar la intrarea în parcare va fi montată o barieră comandată de un servomotor. Accesul unui automobil în parcare este permis prin ridicarea barierei doar dacă mai este cel puțin un loc liber disponibil. La fiecare ridicare a barierei pentru intrare în parcare, numărul de locuri libere disponibile va fi decrementat, iar la fiecare ridicare a barierei pentru ieșire din parcare, numărul de locuri libere va fi incrementat.

• Codul sursă va trebui să ţină cont în redactare de constrângerile specifice care pot apărea în cadrul unui sistem încorporat.

*Descriere sistem:*

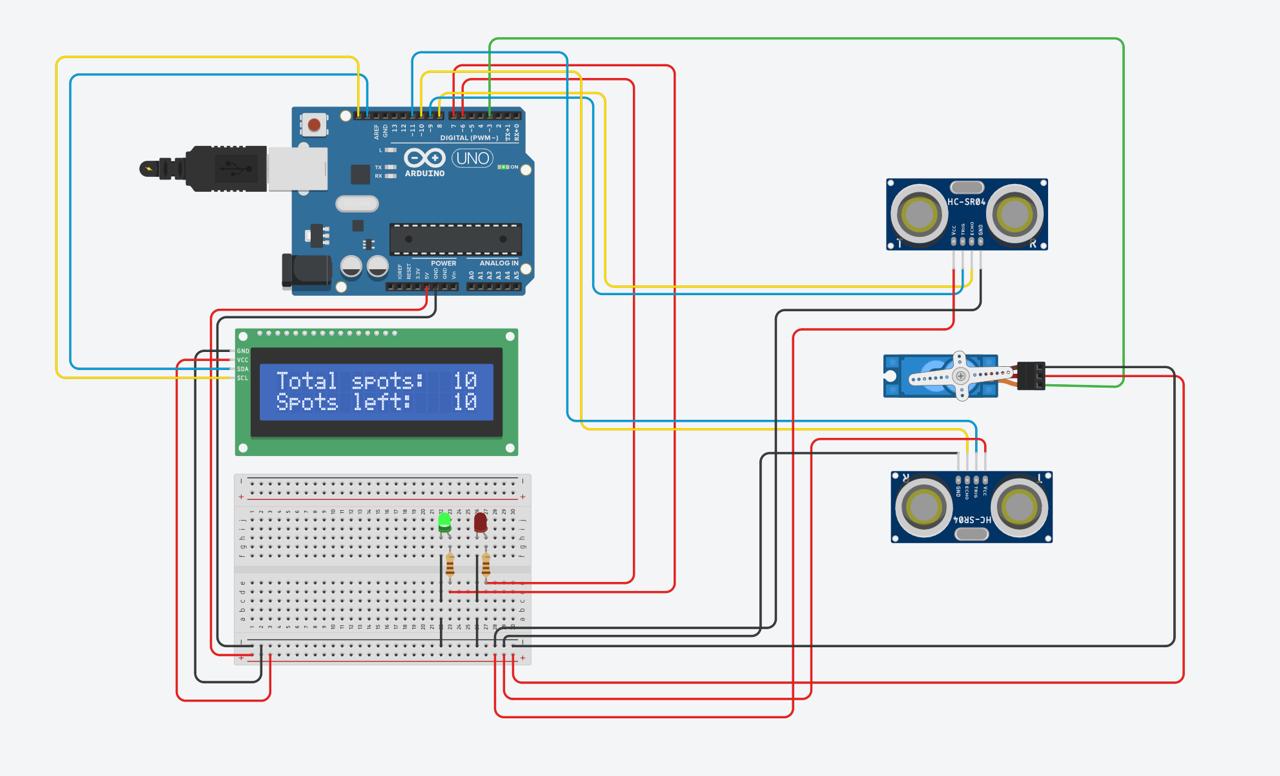
În momentul în care unul dintre senzori este activ, bariera se va ridica, permițând accesul autovehiculului în parcare. Dacă și cel de-al doilea senzor se activează, urmând ca primul să se stingă, înseamna ca mașina a trecut și bariera poate reveni la poziția inițială. Contorul se va modifica în funcție de ce senzor este activat prima dată.

*Descrierea circuitului:*

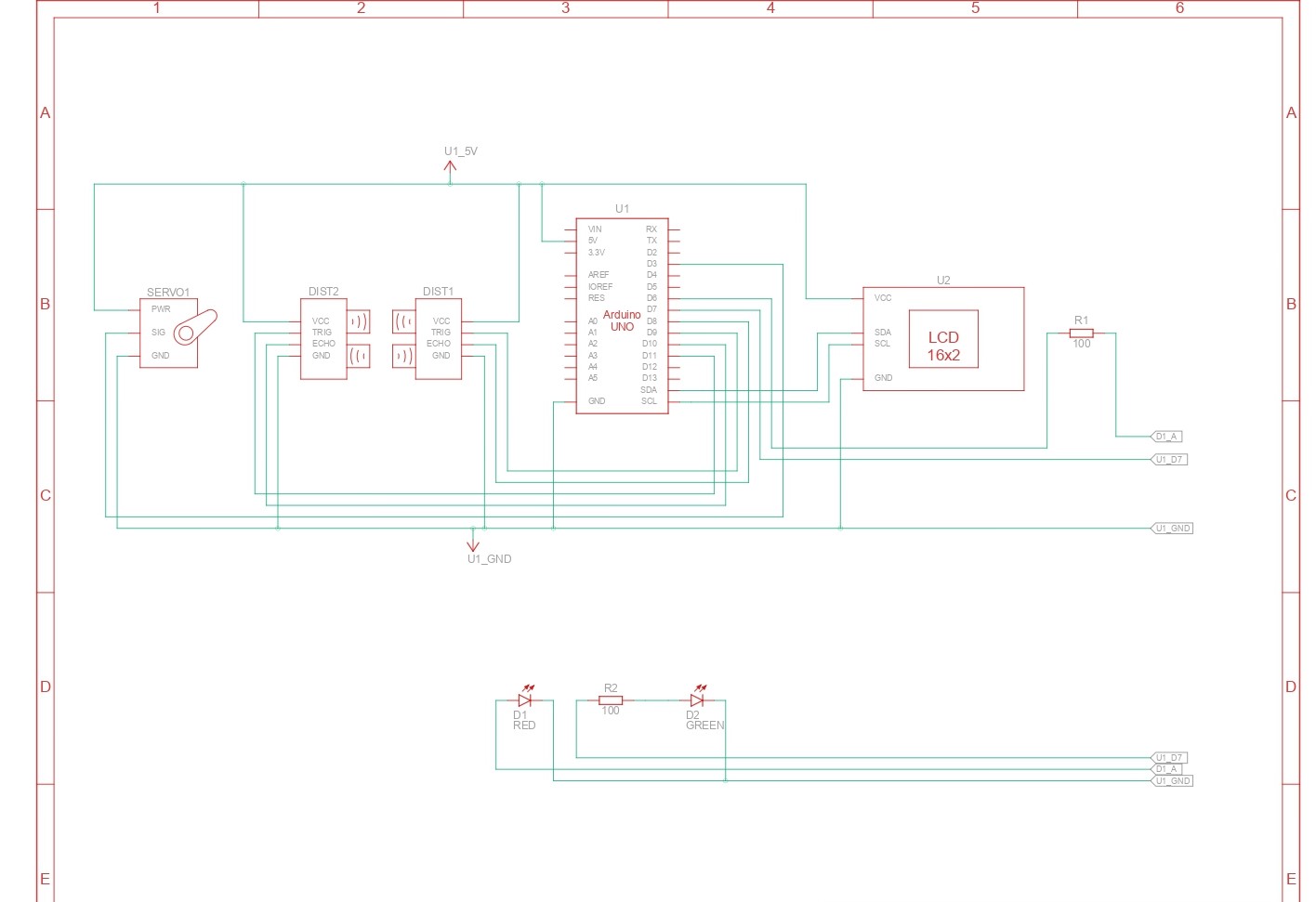
În cadrul dezvoltării circuitului am folosit:

* O placuță Arduino UNO
* Un ecran LCD I2\_C (Liquid Crystal Display)
* Doi senzori cu ultrasunete.
* Un servomotor
* Doua LED-uri

Schema circuitului de mai jos prezintă conexiunile și componentele sistemului de monitorizare a locurilor libere dintr-o parcare, realizată in TinkerCad.

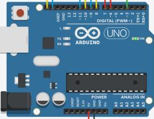


**Schema internă a circuitului:**



*Descrierea componentelor:*

1. **Arduino UNO**



Pinii folosiți in cadrul proiectului sunt:

* GND -pin pentru conectare la masa;
* 5V - tensiune de alimentare reglată și este utilizat pentru a furniza alimentare plăcii, precum și componentelor de pe placă;
* Pin 3,6,7,8,9,10,11- intrări sau ieșiri digitale pentru placa Arduino;

1. **Senzori ultrasunete**



Pinii sunt conectați in următorul mod:

-VCC este conectat la 5V

-GND este conectat la GND

-Trig este conectat la pinul 9, respectiv 11

-Echo este conectat la pinul 8, respectiv 10

**3.Breadboard**

A close-up of a computer

Description automatically generated

Structura breadboard-ului facilitează conectarea componentelor electronice prin inserarea lor în găurile plasate pe matricea de puncte, care sunt interconectate intern conform unui model prestabilit.

4**.Servomotor**



* VCC -pin pentru alimentare
* GND -pin pentru masa
* Signal – conectat la pinul 3

**5.LCD I2\_C 16x2**

A close-up of a computer chip

Description automatically generated

GND-pin pentru masa

VCC-pin pentru alimentare

SDA-pin de date pentru I2C

SCL-pin de clock pentru I2C

Adresa LCD-ului este specificata de catre producator

**6.Led-uri**



Anodul led-urilor este conectat la un rezistor de 100 de ohmi și catodul este conectat la GND.

*Procesul de dezvoltare software:*

Întreg codul proiectului nostru este scris in limbajul de programare C++.

Extensia de fișier “.ino” este asociată cu limbajul de programare Arduino. Fișierele cu extensia .ino sunt folosite în platforma Arduino pentru a scrie și încărca cod pe plăcile Arduino. Acest limbaj de programare este bazat pe C/C++, dar include și biblioteci și funcții specifice pentru a facilita programarea microcontrolerelor Arduino.

In fișierul codului sursă se regăsesc o multitudine de funcții specifice, fiecare dintre ele având un scop bine definit, facilitând rularea și buna funcționare a implementării sistemului.

Codul a fost scris, compilat și rulat prin intermediul aplicației Arduino IDE.

***Configurare Hardware***

* **Funcția setup()**

1. Inițializarea comunicației seriale:



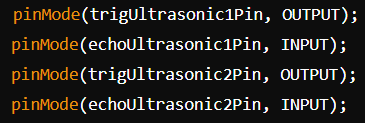
Acest lucru inițializează comunicația serială la o viteză de 9600 bauds, permițând trimiterea și primirea datelor prin portul serial.

1. Atașarea servo-motorului:



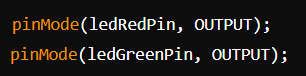
Această comandă atașează servo-motorul la pinul digital 3 al microcontroller-ului

1. Setarea modului pinilor pentru senzorii ultrasonici:



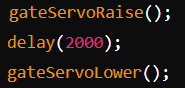
Se setează pinii trigger ai senzorilor ultrasonici (pinul 9 si 11) ca ieșiri și pinii echo (pinul 8 si 10) ca intrări.

1. Setarea modului pinilor pentru LED-uri:



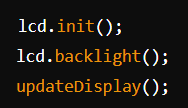
Se setează pinii pentru LED-ul roșu (pinul 6) și pentru LED-ul verde (pinul 7) ca ieșiri.

1. Poziționarea servo-motorului:



Se ridică bariera folosind funcția “gateServoRaise()” (servo-motorul se mișcă la 90 de grade), se așteaptă 2 secunde și apoi se coboară poarta folosind funcția “gateServoLower()” (servo-motorul se mișcă la 0 grade).

1. Inițializarea ecranului LCD:



Se inițializează ecranul LCD cu “lcd.init()” și se pornește iluminarea de fundal cu “lcd.backlight()”. Apoi se actualizează afișajul cu funcția “updateDisplay()”, care afișează numărul total de locuri și locurile disponibile.

1. Setarea stării inițiale a LED-urilor:



LED-ul verde este aprins inițial pentru a indica faptul că sunt locuri disponibile.

Aceste inițializări configurează toate componentele hardware necesare pentru funcționarea corectă a sistemului de parcare descris in cod.

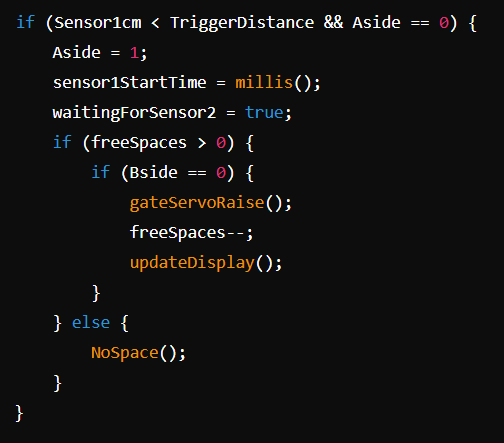
* **Funcția loop()**
  + 1. Măsurarea distanței cu senzorii ultrasonici:





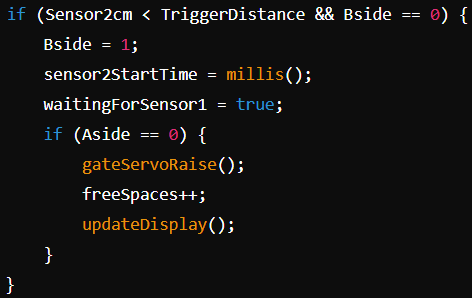
Aceaste funcții sunt apelate pentru a măsura distanța față de un obiect folosind senzorii ultrasonici. Rezultatele sunt stocate în variabilele “Sensor1cm” respectiv “Sensor2cm”.

* + 1. Detectarea prezenței unei mașini la primul senzor:



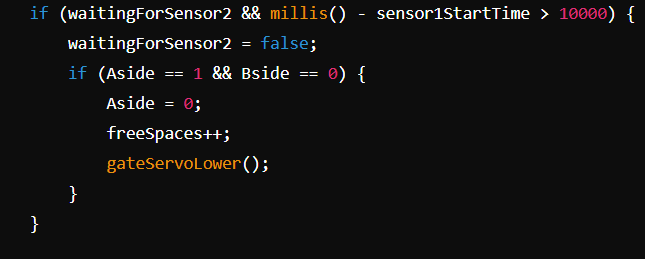
Dacă distanța măsurată de primul senzor este mai mica decât “TriggerDistance” (deci este detectată o mașina) și “Aside” este 0 (adică nu a fost detectată o mașina recent), variabilele “Aside” și “waitingForSensor2” sunt setate pentru a indica prezența mașinii și începutul unui timp de așteptare pentru al doilea senzor. Dacă există locuri disponibile (“freeSpaces > 0”) și al doilea senzor nu detectează nicio mașină (“Bside == 0”), se ridică poarta (“gateServoRaise()”), se decrementează numărul de locuri disponibile și se actualizează afișajul. Dacă nu sunt locuri disppnibile, se afișează mesajul “NO SPACE”.

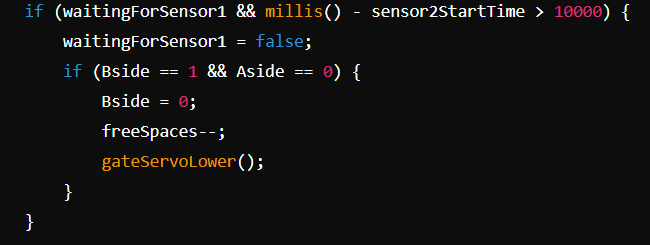
* + 1. Detectarea prezenței unei mașini la al doilea senzor:



Similar cu primul senzor, doar ca în acest caz se incrementează numărul de locuri disponibile.

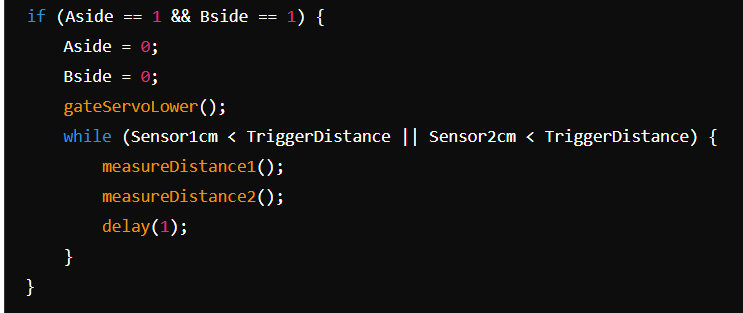
* + 1. Tratarea cazurilor cand mașina nu trece de primul sau al doilea sensor in timp util:





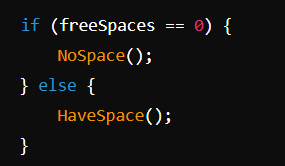
Aceste condiții verifică dacă mașina nu a trecut d eprimul sau al doilea sensor într-un interval de timp prestabilit (10 secunde). Dacă nu a trecut în timp util, barierele laterale sunt coborâte și numărul de locuri disponibile este actualizat corespunzător.

* + 1. Resetarea stării senzorilor și coborârea barierei după detectarea trecerii mașinii:

******

Dacă atât “Aside” cât și “Bside” sunt 1 (mașina a trecut de ambii senzori), atunci ambele variabile se vor reseta la 0, bariera se va coborî și distanțele vor fi măsurate repetat până când mașina a trecut complet de ambii senzori.

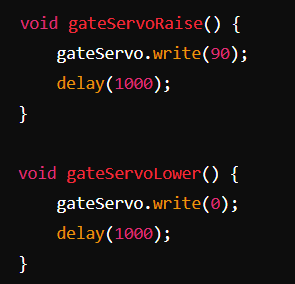
* + 1. Afișarea mesajului corespunzător pe LCD în funcție de locurile disponibile:



Dacă nu ma sunt locuri disponibile (“freeSpaces == 0”), va fi afișat mesajul “NO SPACE”. În caz contrar, se afișează mesajul cu locurile disponibile.

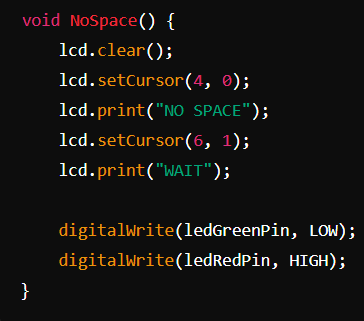
* **Funcții auxiliare**

1. gateServoRaise() și gateServoLower()



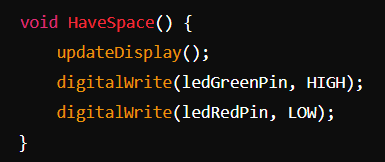
Ridică și coboară bariera.

1. NoSpace()



Afișează mesajul “NO SPACE” și schimbă starea LED-urilor

1. HaveSpace()



Afișează numărul locurilor disponibile și schimbă starea LED-urilor

1. updateDisplay()



Actualizează afișajul LCD cu numărul total de locuri și locurile libere

1. measureDistance1() si measureDistance2()



Măsoară distanțele detectate de cei doi senzori ultrasonici

*Concluzii:*

Implementarea acestui sistem reprezintă o provocare interesantă si captivantă in domeniul dezvoltării software si al ingineriei hardware, oferind o experiență interactivă atât pentru dezvoltatori, cât și pentru utilizatori.

Având în vedere restricțiile de memorie și putere de calcul ale unei plăci Arduino, implementarea trebuie să fie eficientă din punct de vedere al resurselor.

Optimizarea codului și gestionarea atentă a memoriei sunt esențiale pentru a asigura funcționarea corectă și fără probleme a sistemului.

În timpul implementării, testarea riguroasă și depanarea sunt cruciale pentru identificarea și remedierea erorilor și bug-urilor. Folosirea unor tehnici adecvate de testare și depanare ajută la asigurarea fiabilității și stabilității sistemului.

În final, implementarea sistemului pe platforma Arduino reprezintă nu doar o realizare tehnică, ci și o oportunitate de a experimenta și de a învăța mai multe despre dezvoltarea software, ingineria hardware și interacțiunea dintre acestea.

*Întreg codul sursă:*

#include <LiquidCrystal\_I2C.h>

#include <Servo.h>

#include <Wire.h>

LiquidCrystal\_I2C lcd(0x3F, 16, 2);

Servo gateServo;

const int trigUltrasonic1Pin = 9;

const int echoUltrasonic1Pin = 8;

const int trigUltrasonic2Pin = 11;

const int echoUltrasonic2Pin = 10;

const int ledRedPin = 6;

const int ledGreenPin = 7;

const int initialSpots = 7;

int freeSpaces = initialSpots;

int TriggerDistance = 6;

int duration, Sensor1cm, Sensor2cm;

int Aside = 0, Bside = 0;

unsigned long sensor1StartTime = 0, sensor2StartTime = 0;

bool waitingForSensor2 = false, waitingForSensor1 = false;

void setup() {

Serial.begin(9600);

gateServo.attach(3);

pinMode(trigUltrasonic1Pin, OUTPUT);

pinMode(echoUltrasonic1Pin, INPUT);

pinMode(trigUltrasonic2Pin, OUTPUT);

pinMode(echoUltrasonic2Pin, INPUT);

pinMode(ledRedPin, OUTPUT);

pinMode(ledGreenPin, OUTPUT);

gateServoRaise();

delay(2000);

gateServoLower();

lcd.init();

lcd.backlight();

updateDisplay();

digitalWrite(ledGreenPin, HIGH);

}

void loop() {

measureDistance1();

measureDistance2();

if (Sensor1cm < TriggerDistance && Aside == 0) {

Aside = 1;

sensor1StartTime = millis();

waitingForSensor2 = true;

if (freeSpaces > 0) {

if (Bside == 0) {

gateServoRaise();

freeSpaces--;

updateDisplay();

}

} else {

NoSpace();

}

}

if (Sensor2cm < TriggerDistance && Bside == 0) {

Bside = 1;

sensor2StartTime = millis();

waitingForSensor1 = true;

if (Aside == 0) {

gateServoRaise();

freeSpaces++;

updateDisplay();

}

}

if (waitingForSensor2 && millis() - sensor1StartTime > 10000) {

waitingForSensor2 = false;

if (Aside == 1 && Bside == 0) {

Aside = 0;

freeSpaces++;

gateServoLower();

}

}

if (waitingForSensor1 && millis() - sensor2StartTime > 10000) {

waitingForSensor1 = false;

if (Bside == 1 && Aside == 0) {

Bside = 0;

freeSpaces--;

gateServoLower();

}

}

if (Aside == 1 && Bside == 1) {

Aside = 0;

Bside = 0;

gateServoLower();

while (Sensor1cm < TriggerDistance || Sensor2cm < TriggerDistance) {

measureDistance1();

measureDistance2();

delay(1);

}

}

if (freeSpaces == 0) {

NoSpace();

} else {

HaveSpace();

}

}

void gateServoRaise() {

gateServo.write(90);

delay(1000);

}

void gateServoLower() {

gateServo.write(0);

delay(1000);

}

void NoSpace() {

lcd.clear();

lcd.setCursor(4, 0);

lcd.print("NO SPACE");

lcd.setCursor(6, 1);

lcd.print("WAIT");

digitalWrite(ledGreenPin, LOW);

digitalWrite(ledRedPin, HIGH);

}

void HaveSpace() {

updateDisplay();

digitalWrite(ledGreenPin, HIGH);

digitalWrite(ledRedPin, LOW);

}

void updateDisplay() {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Total spots: ");

lcd.setCursor(14, 0);

lcd.print(initialSpots);

lcd.setCursor(0, 1);

lcd.print("Spots left: ");

lcd.setCursor(14, 1);

lcd.print(freeSpaces);

}

void measureDistance1() {

digitalWrite(trigUltrasonic1Pin, LOW);

delayMicroseconds(2);

digitalWrite(trigUltrasonic1Pin, HIGH);

delayMicroseconds(10);

digitalWrite(trigUltrasonic1Pin, LOW);

duration = pulseIn(echoUltrasonic1Pin, HIGH);

Sensor1cm = duration / 29 / 2;

delay(100);

}

void measureDistance2() {

digitalWrite(trigUltrasonic2Pin, LOW);

delayMicroseconds(2);

digitalWrite(trigUltrasonic2Pin, HIGH);

delayMicroseconds(10);

digitalWrite(trigUltrasonic2Pin, LOW);

duration = pulseIn(echoUltrasonic2Pin, HIGH);

Sensor2cm = duration / 29 / 2;

delay(100);

}

*Bibliografie:*

* <https://openlab.citytech.cuny.edu/emtlabs/breadboard/>
* <https://robu.in/arduino-pin-configuration/>
* https://www.tinkercad.com/
* https://www.arduino.cc